CSD101: Introduction to computing and programming (ICP)

Memory in C programs

- Memory in C programs can be divided into 3 conceptual types:
  - Memory for program and data that stays till program is executing. Program itsef, global data, static variables, data in `main`.
  - Stack memory. Memory used when functions are called. This is automatically allocated and retrieved when the function call is over.
  - Dynamic/Heap memory. Memory controlled by the programmer for dynamic data structures.
- A `void*` pointer is a pointer to data of any type and is explicitly coerced to a type when needed. This is required when working with dynamic memory.

Dynamic memory I

- Dynamic memory (or heap memory) is allocated and freed by the programmer.
- Unlike stack memory it is not allocated or deallocated automatically.
- The **C** standard library (`stdlib`), has 4 functions to do this. Each allocation function returns a `void*` pointer to the memory chunk if memory allocation was successful else returns `NULL`.
  - `void *malloc(size_t size)` - size is the size of the memory needed. This is usually specified using the `sizeof` function for the data type for which the memory is needed. Memory is not initialized.
  - `void *calloc(size_t nitems, size_t size)` - nitems is the number of items and size is the size of each item. The memory is initialized to 0.

Dynamic memory II

- `void *realloc(void *ptr, size_t size)` - ptr is a pointer to memory allocated earlier by one of the allocation functions. It is resized to `size`.
- `free(void *ptr)` - deallocates memory earlier allocated by one of the allocation functions.

Dangling pointers and memory leaks

- Dynamic memory allocation can lead to two serious kinds of bugs in programs: dangling pointers and memory leaks.
- Dangling pointers: these are created when a pointer points to allocated memory that has been freed. If we try to use the freed memory the results are undefined.
- Memory leaks: this happens when the only pointer to some allocated memory starts pointing to some other allocated memory of the same type without freeing the first chunk of memory. Now the program has no way to reclaim this memory since there is no reference to it and it is effectively lost. If this happens repeatedly then the program can run out of dynamic memory. A memory leak is the most common bug in C (and C++) programs. It is hard to catch since it can lead to program failure only much later.
- So, always **free** memory when not needed anymore - `free(<ptr>)`.

Dynamic memory use

- Dynamic memory can be used to create variable sized strings, arrays and linked structures (most important use).
- For examples: see code `heap1.c`, `heap2.c`.

Linked structures

- Structure instances can be linked in a chain by using pointers to structures via a self referential mechanism.

```
struct Node {
  int data;
  struct Node *next;
};
// next points to the next structure instance or is NULL
```

- While declaring a `Node` we have to use the *tag* style of declaration and cannot use `typedef`.